The OKS Configuration Framework

Gordon Crone, Eric Flumerfelt, Giovanna Lehmann Miotto

DAQ General Meeting

25 March 2024

What is OKS?

OKS (Object Kernel Support) is a suite of packages originally written for the ATLAS data acquisition effort. Its features include:

- The ability to define object oriented classes which are represented through a schema, used to generate the corresponding c++ / python classes.
- A class in OKS has a class name, basic (int, vector, string,...) and complex (other classes) data members, called respectively "attributes" and "relationships", and methods. OKS classes can also inherit from other classes to extend their functionality.
- A OKS configuration is a collection of instances of one or more OKS classes (“objects”), and is stored in a database.

https://github.com/DUNE-DAQ/dal/blob/develop/docs/README.md

Why OKS?

- Configuration Systems are hard
  - Experience across experiments shows shortcomings of different configuration management systems
- Prototype system using jsonnet/moo sufficient for early operation
  - No schema/configuration editors
  - Re-generate complete configuration to modify parameters
  - Must generate configuration each time, hard to change behavior of generation
- Use pre-existing configuration system
  - Free up effort to work on integration into workflow
- Reduce configuration duplication and allow use of references within configuration
- Load complete configuration and make accessible from C++ code
- OKS has been in use in ATLAS trigger, a similar system to DUNE-DAQ

OKS in DUNE-DAQ: Key Concepts

- Database: A collection of OKS objects. Objects have attributes (configuration values) and relations (references to other objects). Databases are currently stored in XML files
- Schema: A collection of object definitions. Schemas are currently stored in XML files and passed through code-generation (similar to moo)
- File includes: Schemas can include other schemas. Databases can include other databases and schemas. For a Database to be valid, all objects in database must be defined in the included schema tree

OKS in DUNE-DAQ: Description of the DAQ

DUNE-DAQ-specific schemas are stored in the *coredal* and *appdal* repositories. *coredal* schemas describe the structure of the configuration, while *appdal* schemas are concerned with the configuration of individual components.

- Session: Entry point for DUNE-DAQ configurations, defines several top-level configuration parameters
- Segment: Collection of related applications within a Session. Segments can contain other Segments, while Session only has one “root” Segment
- DaqApplication: Represents a single application, with fixed queues, connections, and DAQModules
- SmartDaqApplication: Represents a single application, but generates queues, connections, and modules via an application-specific implementation of a generator function
  - This allows for separation between run-time organization (messaging connections) and configuration of the individual modules

OKS in DUNE-DAQ: More About SmartDaqApplication

- SmartDaqApplication represents a “self-constructing” application through the implementation of a “*generate_modules*” method, which creates a known application structure including modules and messaging queues and network connections
  - *generate_modules* ingests application-specific module configurations (e.g. DFApplication is given a TRB and DataWriter configuration) to produce fully-configured applications
- Current SmartDaqApplication implementations include DFApplication, ReadoutApplication, TriggerApplication, and more
  - These implementations represent all of the applications needed for a basic DUNE-DAQ Session
  - This allows for very static configurations, as run-time messaging links are generated on-the-fly
  - Generation is deterministic for a given configuration, and can be performed outside of the running DAQ context to check output of *generate_modules*

OKS in DUNE-DAQ: Static and Generated Configuration Databases

- A number of database files have been written with parameters which rarely change: https://github.com/DUNE-DAQ/appdal/tree/develop/config/appdal
- Specific test sessions can use these databases to reduce the number of additional objects needed to fully describe the configuration
  - Run-time objects are currently stored in the test/config directory of appdal: https://github.com/DUNE-DAQ/appdal/tree/develop/test/config

OKS In Action: Running the test-session

- Main Documentation page: https://github.com/DUNE-DAQ/oksconfgen/wiki/Setting-up-a-dunedaq-v5.0.0-Development-Area
- Demo
  - Learn about SmartDaqApplications in a session
    - listApps and generate_modules_test
      - listApps test-session sourcecode/appdal/test/config/test-session.data.xml
      - Generate_modules_test test-session df-01 sourcecode/appdal/test/config/test-session.data.xml
  - Use pre-made boot.json to run NanoRC
    - nanorc --partition-number 2 test_config test-session boot conf wait 10 start_run 111 wait 60 stop_run scrap terminate
  - Check output for expected data
    - h5dump -H swtest_run000111_0000_df-01_datawriter-1_*|less
- *drunc*: https://github.com/DUNE-DAQ/drunc/wiki/Testing-the-OKS-configuration

OKS In Action: Generating a Readout Segment

- *oksconfgen* provides scripts (and Python libraries) for generating the most commonly-changed parts of a configuration
  - Replaces some functionality from *daqconf*, but configuration generation will be much more limited in the OKS system
- Demo
  - Using *dromap2oks* to convert existing DRO map
    - dromap2oks DetectorReadoutMap.json
  - Using *generate_readoutOKS* to create Readout session
    - generate_readoutOKS -i config/appdal/fsm.data.xml -s DetectorReadoutMap.data.xml generated-ru-segment.data.xml
  - Using *consolidate_files* to create new Session from existing
    - consolidate_files -i generated-ru-segment.data.xml -i sourcecode/daqsystemtest/integtest/minimal_system_quick_test.data.xml generated-session.data.xml

OKS In Action: Changing a Configuration Parameter

- OKS Databases are XML, can be edited by hand
- But, *dbe* provides GUI editors which keep the databases formatted and sorted, as well as resolving includes to ensure that configuration is complete
- Demo
  - Open generated session file in editor, show objects
  - Open generated session file in *dunedaq_dbe*, show how objects are represented and linked together
- https://dune-daq-sw.readthedocs.io/en/latest/packages/dbe/

OKS In Action: Enabling/Disabling an Object

- A nice feature of OKS is that objects can be disabled at the Session level
  - This allows for “superset” configurations
  - Resource Manager implementation could use enable/disable to generate Session
- Demo
  - Adding object to Session’s disabled relationship in GUI editor
  - Using *oks_enable* to re-enabled disabled object

OKS In Action: Adding Applications to a Segment

- Demo
  - Use existing objects to create an additional DFApplication instance

OKS In Action: Retrieving Configuration Data from Your Code

- DAQModule::init signature changed to give a handle to the OKS objects
  - Retrieve current module object by ID
  - Access attributes of module object
  - Access other parts of the configuration if needed
- Demo
  - https://github.com/DUNE-DAQ/listrev/blob/138c813a4eeac1e31ad5a1a450a2349006e8de93/plugins/ReversedListValidator.cpp#L54
  - https://github.com/DUNE-DAQ/listrev/blob/138c813a4eeac1e31ad5a1a450a2349006e8de93/plugins/ReversedListValidator.cpp#L82
  - https://github.com/DUNE-DAQ/listrev/blob/138c813a4eeac1e31ad5a1a450a2349006e8de93/schema/listrev/listrev.schema.xml#L111
  - (Warning: advanced usage!) https://github.com/DUNE-DAQ/dfmodules/blob/6c787a25bd14e6bc17857e623b25a517c5615025/plugins/TriggerRecordBuilder.cpp#L117

To-Do List

- When dunedaq-v5.0.0 is ready, we would like developers to assist in evaluating the functional state of the system
  - Test provided configurations and give feedback on any difficulties encountered
  - Prepare new, more complex configurations
  - Validate approach to the configuration description
    - SmartDaqApplications ingest configuration objects and create full applications
  - Provide overall feedback on user (un)friendliness
  - Complete transition of remaining DAQModules (ctbmodules, crtmodules, wibmod, sspmodules, daphnemodules)
  - Replicate and test NP04 and NP02 configurations

Questions?

- #daq-oks-integration Slack channel
- Github Issues